Magic Musicplayer - JavaFX in 2019

Blog - Programming - 01

Luca Hohmann
business@lucahohmann.com

02. April 2019

Abstract

Today we are finally going over a project of mine. My in development music player "Magic MusicPlayer". I'll start with a short introduction of what it is and what I want to achieve with it. Then I jump right into the development. What works? What is planned? Where can you download it? In the end I'll talk a bit about why I chose Java and JavaFX, as well as some problems with this decision.

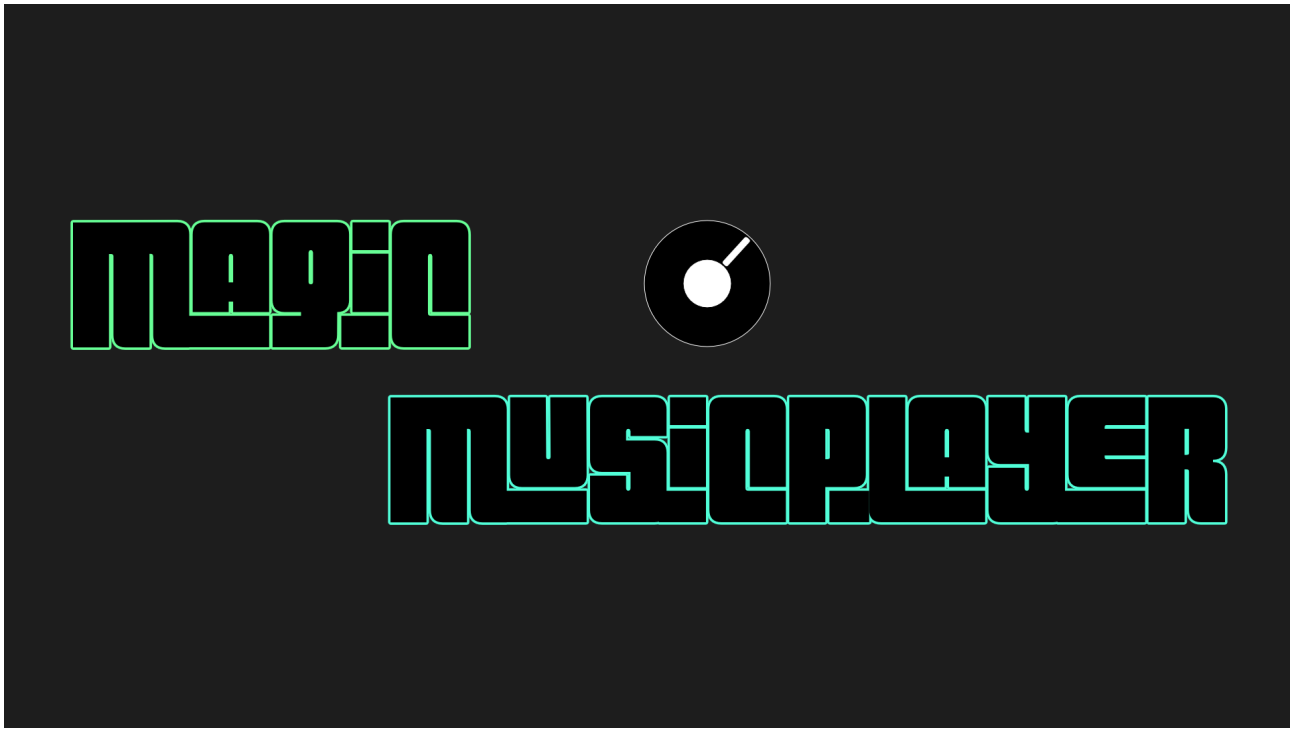

Figure 0.1 Musicplayer Logo

1 Introduction

1.1 What is the Magic Musicplayer?

As the name suggests, it is a music player. At the moment I focus on releasing it on Windows, but anyone who can compile it him/herself, can get it on Linux or MacOS as well. It is completely Open Source, so everyone can contribute, take a look into the code, and just change things he/she dislikes.

1.2 What is my goal?

There are different aspects pushing me to program my own musi cplayer application. At the beginning there were 2 reasons for me to do this. The first one being that no music player for local music files does the ordering of artists in the only reasonable way I could think of. And that is, when a song is created by 2 or more artists it should be added to both artist pages rather than creating a new one which is named after the combination of those two artist. I'll come back to that later in "The development". The second reason I wanted and still want to create such an application is, that at university, we mostly work on weekly projects or projects which last one to three months. In rare cases a project spans over the course of a whole semester. And even then, most of these projects are for learning purposes only, and do not have that much real world usage. And yes this is how university should be. It should educate. But the actual process of sticking longer to a project cannot be taught, just experienced. So here I am.

One of the biggest aspects developed within the last months, as I am becoming more aware of privacy issues with applications. As a result I care more about using safe Open Source alternatives to closed source software of big companies. So as I'm currently not in the position to provide actual functionality to existing projects, because these are often already very big and need functionality, which I'm not capable of, I just provide an application myself.

2 The development

2.1 What is available today?

At the time of writing, these basic features are already available:

1. Crawling music files
2. GUI
3. Playback of songs
4. Persistent album, artist and song storage
5. Display of metadata
6. Scrolling through songs
7. Playback
8. Association of songs to artists and albums

All of these features provide basic functionality to any music player. Most of them are rather easy to implement, but need a good foundation, and a clear assignment of tasks to classes. So it took rather long to get to this point of development. The project started during the last semester holidays, besides another project, which is currently on ice. Within the holidays I got the foundation and a basic GUI system working. The GUI system is actually the hardest part of the development, as many things need tricks and tweaking to work. But more on the JavaFX GUI system in "Java and JavaFX - a retrospect". During the semester I worked very infrequent on it, as I was busy with work from university, my job and events we attended. But how does one organize such a project, which consists of many different areas?

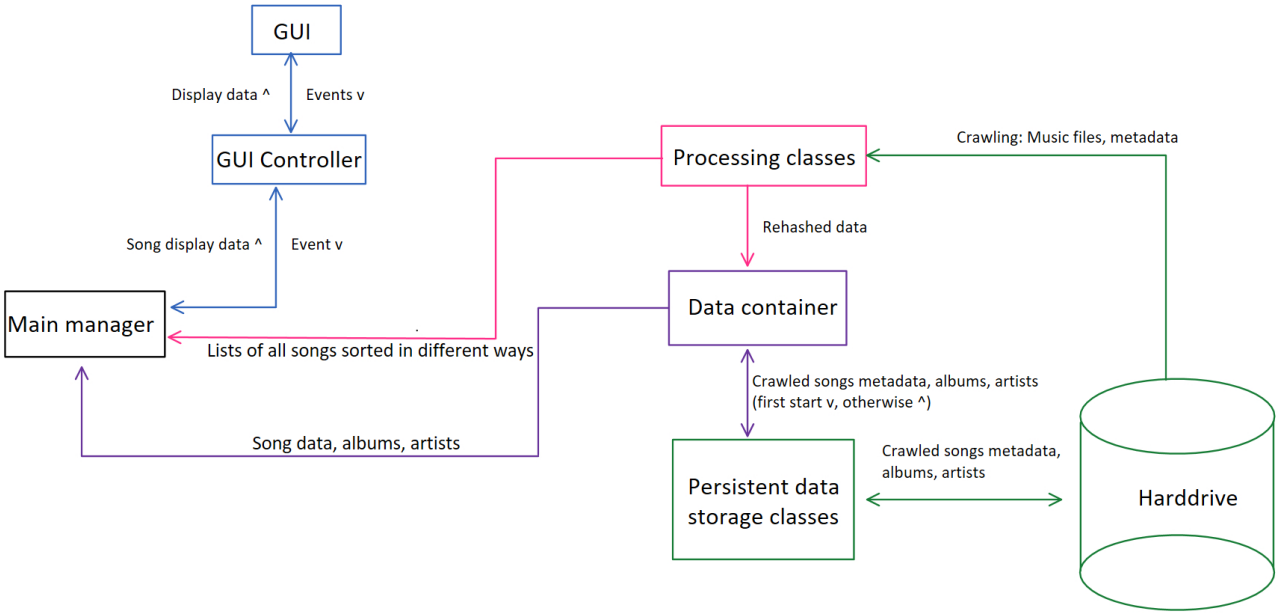

Figure 2.1 Rough visualization of project structure (data flow) - ^v represent corresponding directions

Figure 2.1 shows the rough structure of the project with the data flow between task groups. Call wise, the main manager handles most of the classes. It starts all processing and storage threads. The processing classes are responsible for crawling the given directories for music files and cover images. Then all of the found files are processed and put into separate instances of a music file class, which holds all metadata as well as the path to the actual file (part of the Data container classes). Then the single music files are combined into list of their respective albums and artists. These lists are then send to the main manager for usage.

While the user scrolls through his/her songs, listens to them and wonders why some things do not work yet (still alpha build ;), the persistent storage classes are getting to work, with the music files, albums and artists. Each of the three get their own file, where all data from the corresponding lists of data containers are saved in a simple, yet effective, format. In my test scenario the only advantage of persistent saving is, that I do not have to wait until it has crawled all my music files again (>1500). The album and artist combinations work so fast, that it is done before I would have the chance to change to the album or artist view (not implemented yet). As mentioned before, I really dislike the fact, that local music players often have one big issue in common. And that is the inability to distinguish multiple artists from the tags of a file. On the one hand this is due to the fact, how multiple artists are mentioned. Some are named in the title, e.g. songtitle (ft. secondartist), others are written into the tags, but then artists are not that easily distinguishable. That comes from the format they are stored in:
artist1/artist2/artist4...
This format is also not consistent across all files, and depends on which tags include additional artists and which not. That makes it very hard to automate this process. Let alone the different versions of the mp3 tags that are used. There is Id3v1 and Id3v2, as well as custom formats. Id3v1 should have been deprecated a long time ago, since it is not capable of allowing as many tags as Id3v2. But somehow, there are still music files, which come with this format. So without a database, which provides a lot of artists (like the one Spotify has), it is nearly impossible, to arrange the artists in the way I'd like to. Of course, I could say "Every song that does not conform to the correct Id3v2 format simply gets a combined artist", but even tho this will b my approach, it is still unsatisfying.

2.2 What is planned?

In order to provide a rich experience, and convince user of this application, some more features are planned for the future. Figure 2.2 gives a good overview of what is planned for the different stages and where I currently stand. Of course, not everything is included, and some priorities might change within

development. A good example for this is the goal to build the application as executable file. With this feature coming late in development, only a few would attempt to build the application him/herself or use it with the .class files. Therefore this is in development at the moment of writing. When it is implemented completely, the download link will be available on Github as well as the project page.

To summarize everything that is currently planned, the features can be categorized:

Support all major file formats; Provide a complete and working GUI; Implement major, as well as modern and interesting functionality.

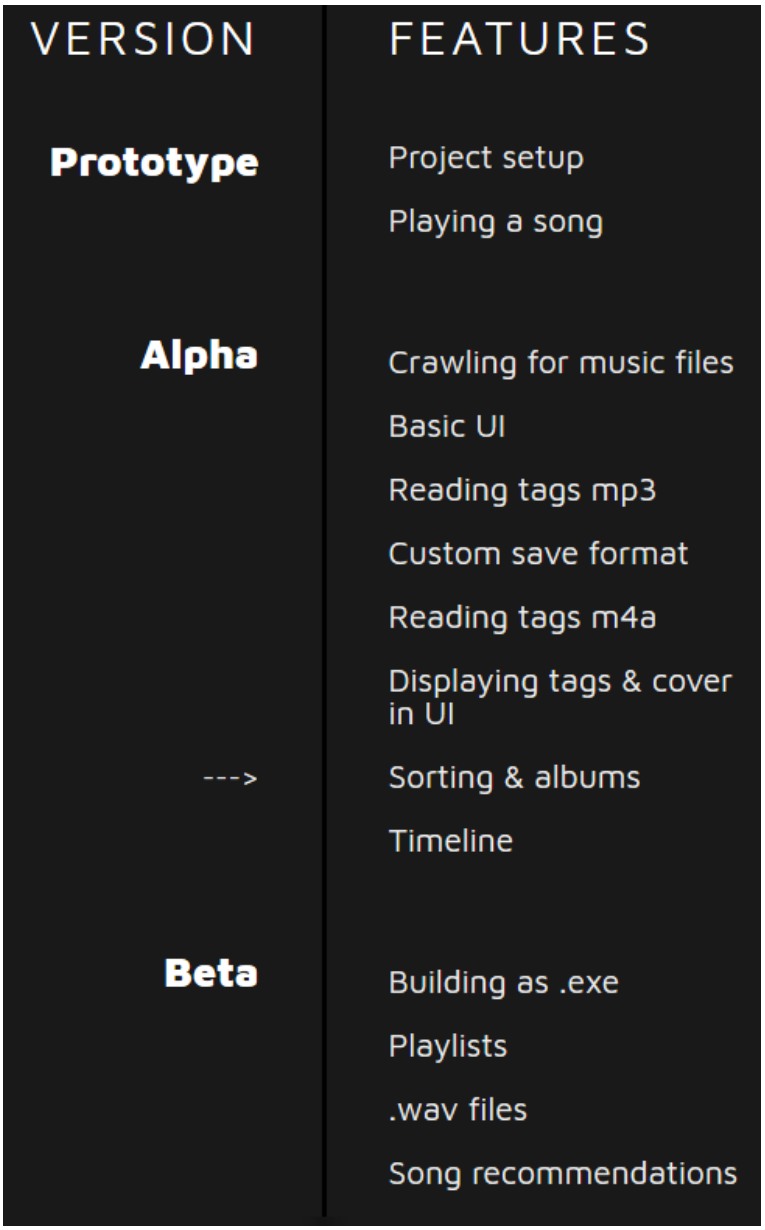

Figure 2.2 Roadmap of the project (Source: lucahohmann.com)

The first two categories are self explanatory and definitely reasonable for every music application. The last category however is a very broad one. Every user has his/her own opinion about what should be available. But at this point I have to be very pragmatic and disillusion possible users. Not every feature seen in music players like Spotify, Deezer and all the others, will be available in a local and offline music player. That is simply not possible, especially since I am the only person working on it. Features like song recommendations, mixes, presenting songs, which have not been listened to for a long time and similar are on my list. But at the moment I cannot estimate how much of an obstacle they will be and therefore not make any promises on them. If someone wants to add features, he/she is invited to do so, and add it to the repository after I looked over it. Furthermore, I want to finish this project within this semester, in a best case scenario, within these holidays (so end of april).

3 Java and JavaFX - a retrospect

At the beginning, I already teased a little bit about JavaFX and Java as a development platform. There are numerous advantages and disadvantages to them, compared to other possible platforms. But I want to go over a few aspects, which have affected me the the most during the development so far.

3.1 Advantages of JavaFX

The biggest advantage is the **simple entrance into JavaFX**. If you are familiar with Java or Java-like programming languages, it will be an ease to get a basic window with UI running. The connection between the backend code and the frontend is very simple through FXML. That is, because it is very well documented and a lot of tutorials on JavaFX have been published by different blogs and websites.

Another good aspect is, that JavaFX provides very good **abstractions for media files**. No other language I am capable of, provided such an easy way to play a mp3 file, or show a video. You create a media player, register a media view in the GUI, and set the media of the player. That's it. Everything else is done by Java and as someone, who is not interested in reinventing the wheel on such simple things, it is great.

3.2 Disadvantages of JavaFX

On the other hand, the biggest disadvantage of JavaFX, is the very **inconsistent UI adjustment**. When I tried to get a resizeable window working, everything broke. It took multiple days and restarts to realize, that it is a complete waste of time. So as a consequence I dumped that, and made the window size fixed. The problem is, that a scaling UI is only possible with certain constellations of UI elements and a lot of manual coding to check whether things are still in place and in correct size. It is also very inconsistent, as sometimes said scaling UI elements do not have the intended effects or are completely ignored by the elements put onto them. This leads to state of frustration quite fast. But I have to mention, that this might be due to the fact that I used JavaFX 2 with Java 8, but more on that in the next argument.

The other bad aspect of JavaFX is the fact, that it is **not backwards compatible**, in contrast to Java. This reason behind this is, that JavaFX is no longer a project of Oracle, but rather a project of Gluon, the providers of Scenebuilder for JavaFX (which I can 100 percent recommend to use). So the problems with JavaFX, such as performance problems in JavaFX 2 for Java SE 8 (the last version from Oracle), inconsistent UI adjustments are not going to be fixed anymore. But changing to Gluons version is not possible, as some things are not yet implemented. So for example, there are still errors with media views, which prohibit using my implementation to play back music.

4 Final words

Now everyone should have a good insight into my current solo project. If yo uwant to take alook at the code or my coding style, you can do so on Github. Please be aware, that I'm currently restructuring the music manager class, so it is not completely in line with my normal coding and commenting conventions.

Something information regarding the blog. Comments are still not available, as I still have problem with the Staticman API. I'll post an issue about this on their github site, when I find the time for it. So stay excited for them.

Do not forget to check out my other projects on my website lucahohmann.com/projects and follow me on Twitter for constant updates, as well as my Substance Alchemist Material challenge, which is currently running. @lucahohmann
And yes, thanks for reading have a good day and See you next time
Luca =)